(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平11－224184

(43)公開日　平成11年(1999)８月17日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/06	５３０	Ｇ０６Ｆ　9/06	５３０Ｖ
12/00	５１３	12/00	５１３Ｄ
17/30		15/40	３８０Ｅ

審査請求　有　　請求項の数９　ＯＬ　（全 24 頁）

(21)出願番号　特願平10－24459

(22)出願日　平成10年(1998)２月５日

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　北野　拓哉
東京都港区芝五丁目７番１号　日本電気株式会社内

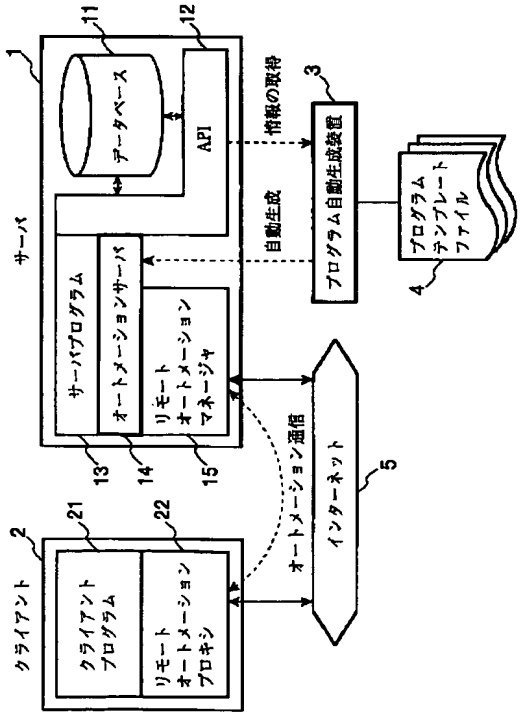
(74)代理人　弁理士　古澤　聡　（外１名）

(54)【発明の名称】　オブジェクト指向データベース操作プログラム生成システム及び方法

(57)【要約】

【課題】　既存のオブジェクト指向データベースを、既存のオートメーション通信の機能を利用して操作するプログラムを自動的に生成する。また、プログラムの開発・保守を容易にする。

【解決手段】　プログラム自動生成装置３は、データベース１１からＡＰＩ１２を介してそのデータベース１１に定義されているスキーマ情報を取得し、取得したスキーマ情報とプログラムテンプレートファイル４を基にして、オートメーション通信のためのオートメーションサーバ１４のためのプログラムを自動生成する。オートメーションサーバ１４は、データベース１１のスキーマを構成するクラスとその属性、メソッドを、オートメーション通信を利用するクライアントプログラム２１に公開する。



1

【特許請求の範囲】

【請求項１】オブジェクト指向データベースに定義されているスキーマを構成する各クラスに関する情報を取得する取得手段と、

前記スキーマを構成する各クラスに含まれるオブジェクトの操作に用いられるプログラムを生成するための第１のテンプレートを記憶する第１のテンプレート記憶手段と、

前記第１のテンプレート記憶手段から前記第１のテンプレートを読み出し、該読み出した第１のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、クラスに含まれるオブジェクトを操作するための第１のプログラムを生成するプログラム生成手段と、

前記プログラム生成手段が生成する前記スキーマを構成するクラス毎のプログラムが外部に公開すべきインタフェースを定義するためのインタフェース定義の生成に用いられる第２のテンプレートを記憶する第２のテンプレート記憶手段と、

前記第２のテンプレート記憶手段から第２のテンプレートを読み出し、該読み出した第２のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記プログラムのインタフェース定義を生成するインタフェース定義生成手段と、を備えることを特徴とするオブジェクト指向データベース操作プログラム生成システム。

【請求項２】前記プログラム生成手段が生成した前記第１のプログラムと前記インタフェース定義生成手段が生成した前記インタフェース定義とをリンクすることによって、実行可能なモジュールを作り上げるリンク手段と、をさらに備えることを特徴とする請求項１に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項３】前記スキーマを構成する各クラスに含まれるオブジェクトを外部から操作するための外部操作情報を生成するための第３のテンプレートを記憶する第３のテンプレート記憶手段と、

前記第３のテンプレート記憶手段から　第３のテンプレートを読み出し、該読み出した第３のテンプレートに、前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記外部操作情報を生成する外部操作情報生成手段と、をさらに備えることを特徴とする請求項１または２に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項４】設定された情報によってインタフェースを介して外部から他のプログラムを操作するための第２のプログラムを実行させるプログラム実行手段と、

前記外部操作情報生成手段が生成した外部操作情報を前記第２のプログラムに設定することによって、前記第２のプログラムが前記インタフェース定義手段によって定義されたインタフェース定義に対応するインタフェースを介して前記プログラム生成手段によって生成された前記第１のプログラムを操作することが可能となるように設定する設定手段と、を備えることを特徴とする請求項３に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項５】前記プログラム生成手段は、

前記スキーマを構成する各クラス毎に、プロパティの取得関数を生成する手段と、プロパティの設定関数を生成する手段と、メソッドを生成する手段とを備えることを特徴とする請求項１乃至４のいずれか１項に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項６】前記インタフェース定義によって定義されるインタフェースは、オートメーション通信を行うためのインタフェースであり、

前記外部操作情報は、オートメーション通信の機能を利用して前記プログラム生成手段が生成した前記第１のプログラムを操作するための情報であることを特徴とする請求項１乃至５のいずれか１項に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項７】前記スキーマを構成するクラスに含まれるオブジェクトは、それぞれ外部から呼び出される場合のインタフェースとなる引数の個数が異なるメソッドを複数定義しており、

前記インタフェース定義は、前記オブジェクトのメソッドが有する引数の最大個数に対応する引数と、前記オブジェクトのメソッドが有する引数の最小個数に対応する引数の省略指定とによって定義されることを特徴とする請求項１乃至６のいずれか１項に記載のオブジェクト指向データベース操作プログラム生成システム。

【請求項８】オブジェクト指向データベースに定義されているスキーマを構成する各クラスに関する情報を取得する取得ステップと、

予め用意された第１をテンプレートを読み出し、該読み出した第１のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、クラスに含まれるオブジェクトを操作するためのプログラムを生成するプログラム生成ステップと、

予め用意された第２のテンプレートを読み出し、該読み出した第２のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記プログラムのインタフェース定義を生成するインタフェース定義生成ステップと、

インタフェース定義生成ステップと、を含むことを特徴とするオブジェクト指向データベース操作プログラム生成方法。

【請求項９】予め用意された第３のテンプレートを読み出し、該読み出した第３のテンプレートに、前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記外部操作情報を生成する外部操作情報生成ステップをさらに含むことを特徴とする請求項８に記載のオブジェクト指向データベース操作プログラム生成方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、オブジェクト指向データベースを操作するためのプログラム、特にオートメーションの機能を利用してオブジェクト指向データベースを操作するためのプログラムを生成するオブジェクト指向データベース操作プログラム生成システム及び方法に関する。

【０００２】

【従来の技術】インターフェース定義や分散ミドルウェアのクラスライブラリを用いたオブジェクトの開発は、分散システムを構築するための特有のプログラミング作業となる。この部分のプログラムを自動生成するアプローチ、或いはこの部分のプログラムの生成を支援しようとするアプローチは、従来よりかなりの数が提案されている。

【０００３】このようなアプローチとして、まず、特開平４－１１１０２２号公報に記載のインタフェースプログラム自動生成装置（以下、従来例１という）が挙げられる。従来例１は、インタフェース定義、開発言語仕様、計算機種別データ表現の一般化／特殊化プログラムルーチン、通信プロトコル別プログラムルーチンの４つの情報源を基に、ネットワーク通信のための交換データ型定義とインタフェースプログラムとを自動生成する。

【０００４】また、上記のようなアプローチとして、特開平８－３６４８６号公報に記載の分散プログラム記述支援装置（以下、従来例２という）がある。従来例２は、特にＣＯＲＢＡに関して、開発言語によるオブジェクト記述から、構文解析、内部表現変換を経て、分散オブジェクトのインタフェース記述とインプリメンテーションとを記述する。

【０００５】さらに、上記のようなアプローチとして、特開平８－２６３２７７号公報に記載のデータ操作プログラムの自動生成装置（以下、従来例３）がある。従来例３は、データベース定義からデータの属性を解析する属性解析部の解析結果、及び型変換規則、データベースアクセステンプレート、補助関数生成規則に基づいて、既存の関係データベースのデータ操作を行うためのオブジェクト指向プログラムを生成する。

【０００６】上記のようなアプローチに従った実際の製品として、Visual C++（登録商標）にツールとして含まれるDeveloper Studioの機能であるAppWizard及びClass Wizard（いずれも商標）（以下、従来例４という）、並びにオブジェクト指向データベース管理システムであるObjectStore（登録商標）の拡張機能であるPSE for ActiveX（以下、従来例５という）がある。

【０００７】従来例４では、AppWizardが開発の開始時点での初期設定入力からオブジェクト記述言語（以下、ＯＤＬという）のインタフェース定義とオートメーションオブジェクト（オートメーションによって操作可能なオブジェクトをいう。以下、同じ）の初版プログラムを自動生成する。ClassWizardがその後の開発言語によるプログラムの開発と並行して、ＯＤＬの記述とオートメーションオブジェクトのプログラムとを自動的に追加、削除、修正していく。

【０００８】従来例５では、ObjectStoreが最初から持っているクラスライブラリのODL記述と、開発者が定義するObjectStore型記述ファイル(拡張子をostとして保存したファイル：以下、ostファイルという)とを合わせて、ObjectStoreの専用コンパイラでアプリケーションのODL記述と対応するオートメーションオブジェクトのプログラムとを生成する。

【０００９】

【発明が解決しようとする課題】しかしながら、上記の従来例には、次のような問題点があった。上記従来例１では、ネットワーク通信のための交換データ型定義とインターフェースプログラムとを自動生成するものの、インタフェース定義自体は、ユーザが与えなければならなかった。また、従来例１の装置は、既存のシステムを利用して構成されるものでないため、実際の製品を開発するためには、かなりの開発期間を要するという問題があった。しかも、インタフェース仕様が変更された場合には、プログラムをコンパイラの構文解析から作り直さなければならず、保守作業にも多くの時間を費やすこととなる問題があった。

【００１０】上記従来例２は、単一のオブジェクト指向プログラミング言語によるオブジェクト記述から、ＣＯＲＢＡ仕様の分散オブジェクトのインタフェース記述とインプリメンテーション記述(ユーザ仕様言語)とを自動生成する機能を備える前提にしている。すなわち、オブジェクト変換プリプロセッサの構成に関するものであり、プログラムそのものを自動的に生成するものではない。しかも、インタフェース仕様が変更された場合には、プログラムをコンパイラの構文解析から作り直さなければならず、保守作業にも多くの時間を費やすこととなる問題があった。

【００１１】上記従来例３では、データベースアクセステンプレートに従って生成されるプログラムは、Ｃ＋＋などの特定の言語に依存するものとなる。このため、例

えば、市販のアプリケーションプログラムが有するオートメーション通信機能を利用したツールを生かし切ることができないという問題点があった。また、インターフェース定義を自動的に生成するものではなく、インタフェース仕様が変更されて場合には、一からプログラムを作り直さなければならず、保守作業にも多くの時間を費やすこととなる問題があった。

【００１２】上記従来例４では、ClassWizardの機能は、AppWizardによって初版プログラムが生成されたプログラムに対してのみ、有効に働くようになっている。このため、AppWizardを使って生成していないプログラムに対して、ClassWizardの機能を十分に使うことができないという問題点があった。

【００１３】上記従来例５では、ostファイルを作成するという別のインタフェース定義を行わなければならず、インタフェース定義そのものを自動生成することができない。このため、従来例３と同様に、市販のアプリケーションプログラムが有するオートメーション通信機能を利用したツールを生かし切ることができないという問題点や、インタフェース仕様が変更されて場合には、一からプログラムを作り直さなければならないという問題点があった。

【００１４】さらに、データベース管理システムやＡＰＩ（Application Programing Interface）や、オブジェクト指向言語で書かれたサーバプログラムがオブジェクトの多相性をサポートしていても、クライアントからオートメーションを利用してアクセスした場合に、その多相性の機能を利用できる技術は、従来存在しなかった。また、既存のオートメーションの機能を利用してデータベースを操作できるようにするプログラムを生成する技術は、従来存在しなかった。

【００１５】本発明は、また、開発・保守が容易なオブジェクト指向データベースを操作するためのプログラムを生成することができるオブジェクト指向データベース操作プログラム生成システム及び方法を提供することを目的とする。

【００１６】本発明は、上記従来例の問題点を解消するためになされたものであり、既存のオブジェクト指向データベースを既存のオートメーション通信の機能を利用して操作するプログラムを、自動的に生成するオブジェクト指向データベース操作プログラム生成システム及び方法を提供することを目的とする。

【００１７】本発明は、また、オブジェクトの多相性の機能を十分に利用することができるオブジェクト指向データベースを操作するためのプログラムを生成するオブジェクト指向データベース操作プログラム生成システム及び方法を提供することを目的とする。

【００１８】

【課題を解決するための手段】上記目的を達成するため、本発明の第１の観点にかかるオブジェクト指向データベース操作プログラム生成システムは、オブジェクト指向データベースに定義されているスキーマを構成する各クラスに関する情報を取得する取得手段と、前記スキーマを構成する各クラスに含まれるオブジェクトの操作に用いられるプログラムを生成するための第１のテンプレートを記憶する第１のテンプレート記憶手段と、前記第１のテンプレート記憶手段から前記第１のテンプレートを読み出し、該読み出した第１のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、クラスに含まれるオブジェクトを操作するための第１のプログラムを生成するプログラム生成手段と、前記プログラム生成手段が生成する前記スキーマを構成するクラス毎のプログラムが外部に公開すべきインタフェースを定義するためのインタフェース定義の生成に用いられる第２のテンプレートを記憶する第２のテンプレート記憶手段と、前記第２のテンプレート記憶手段から第２のテンプレートを読み出し、該読み出した第２のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記プログラムのインタフェース定義を生成するインタフェース定義生成手段と、を備えることを特徴とする。

【００１９】上記のオブジェクト指向データベース操作プログラム生成システムは、前記スキーマをクラスに含まれるオブジェクトを操作するための第１のプログラムと、インターフェース定義とのいずれをも自動的に生成することができる。このため、開発者がプログラムとインタフェース定義とのいずれか一方を与えたり、プログラムとインタフェース定義とを別々のシステムを用いて生成したりする必要がないので、オブジェクト指向データベースを操作するためのプログラムの開発が容易になる。また、インタフェース仕様の変更の場合にも、第１のプログラムを一から作り直さなければならないということがないので、プログラムの保守も容易である。

【００２０】上記のオブジェクト指向データベース操作プログラム生成システムは、前記プログラム生成手段が生成した前記第１のプログラムと前記インタフェース定義生成手段が生成した前記インタフェース定義とをリンクすることによって、実行可能なモジュールを作り上げるリンク手段と、をさらに備えてもよい。

【００２１】上記のオブジェクト指向データベース操作プログラム生成システムは、前記スキーマを構成する各クラスに含まれるオブジェクトを外部から操作するための外部操作情報を生成するための第３のテンプレートを記憶する第３のテンプレート記憶手段と、前記第３のテンプレート記憶手段から　第３のテンプレートを読み出し、該読み出した第３のテンプレートに、前記取得手段が取得したクラスに関する情報のうちの対応するものの

少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記外部操作情報を生成する外部操作情報生成手段と、をさらに備えるものとしてもよい。

【００２２】この場合、上記のオブジェクト指向データベース操作システムは、設定された情報によってインタフェースを介して外部から他のプログラムを操作するための第２のプログラムを実行させるプログラム実行手段と、前記外部操作情報生成手段が生成した外部操作情報を前記第２のプログラムに設定することによって、前記第２のプログラムが前記インタフェース定義手段によって定義されたインタフェース定義に対応するインタフェースを介して前記プログラム生成手段によって生成された前記第１のプログラムを操作することが可能となるように設定する設定手段と、を備えるものとすることができる。

【００２３】これにより、上記のオブジェクト指向データベース操作プログラム生成システムは、例えば、インターネットなどを介して遠隔地から操作指令を送るようにしたクライアントサーバ型のシステム構成とする場合に、クライアント側で使用される第２のプログラムも、既存のプログラムを用いて容易に生成することができる。

【００２４】上記オブジェクト指向データベース操作プログラム生成システムにおいて、前記プログラム生成手段は、例えば、前記スキーマを構成する各クラス毎に、プロパティの取得関数を生成する手段と、プロパティの設定関数を生成する手段と、メソッドを生成する手段とを備えるものとすることができる。

【００２５】上記オブジェクト指向データベース操作プログラム生成システムにおいて、前記インタフェース定義によって定義されるインタフェースは、例えば、オートメーション通信を行うためのインタフェースとすることができる。この場合、前記外部操作情報は、オートメーション通信の機能を利用して前記プログラム生成手段が生成した前記第１のプログラムを操作するための情報とすることができる。

【００２６】このような構成としたことにより、上記オブジェクト指向データベース操作プログラム生成システムは、既存のオートメーション通信のシステムを利用することができる。これにより、プログラムの開発、運用及び保守が容易になる。なお、オートメーション通信は、インターネットなどの通信回線を介して実現する場合に限らず、プロセス間通信の手法によって実現してもよい。

【００２７】上記オブジェクト指向データベース操作プログラム生成システムにおいて、前記スキーマを構成するクラスに含まれるオブジェクトは、それぞれ外部から呼び出される場合のインタフェースとなる引数の個数が異なるメソッドを複数定義しているものであってもよい。この場合、前記インタフェース定義は、前記オブジェクトのメソッドが有する引数の最大個数に対応する引数と、前記オブジェクトのメソッドが有する引数の最小個数に対応する引数の省略指定とによって定義されるものとすることを好適とする。

【００２８】これにより、前記スキーマを構成するクラスに含まれるオブジェクトが多相性をサポートしている場合に、前記オブジェクトのメソッドに引数を渡すためのインタフェース仕様を統一することができるので、例えば、オートメーション通信を利用したような場合にも、オブジェクトの多相性の機能を利用することができる。

【００２９】上記目的を達成するため、本発明の第２の観点にかかるオブジェクト指向データベース操作プログラム生成方法は、オブジェクト指向データベースに定義されているスキーマを構成する各クラスに関する情報を取得する取得ステップと、予め用意された第１をテンプレートを読み出し、該読み出した第１のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、クラスに含まれるオブジェクトを操作するためのプログラムを生成するプログラム生成ステップと、予め用意された第２のテンプレートを読み出し、該読み出した第２のテンプレートに前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記プログラムのインタフェース定義を生成するインタフェース定義生成ステップと、インタフェース定義生成ステップと、を含むことを特徴とする。

【００３０】上記のオブジェクト指向データベース操作プログラム生成方法は、予め用意された第３のテンプレートを読み出し、該読み出した第３のテンプレートに、前記取得手段が取得したクラスに関する情報のうちの対応するものの少なくとも一部を挿入することによって、前記スキーマを構成するクラス毎に、対応する前記外部操作情報を生成する外部操作情報生成ステップをさらに含むものとしてもよい。

【００３１】

【発明の実施の形態】以下、添付図面を参照して、本発明の実施の形態について説明する。

【００３２】図１は、この実施の形態にかかるデータベースアプリケーションプログラムの開発・実行環境を実現するためのシステム構成例を示す図である。

【００３３】図示するように、このシステムは、主としてサーバ１と、クライアント２と、プログラム自動生成装置３とを備える。サーバ１とクライアント２とは、インターネット５を介してオートメーション通信を行う。サーバ１とプログラム自動生成装置３とは、ＬＡＮ回線などの何らかの通信回線を介してデータ通信を行う。プ

ログラム自動生成装置３は、プログラムテンプレートファイル４を参照する機能を有する。

【００３４】サーバ１は、プロセッサ、記憶装置及び入出力装置を有する情報処理装置によって構成され、データベース１１と、ＡＰＩ１２と、サーバプログラム１３と、オートメーションサーバ１４と、リモートオートメーションマネージャ１５との機能を含む。

【００３５】データベース１１は、オブジェクト指向データベースであり、クラス、クラスの属性、クラスのメソッドのそれぞれの名前と型との情報でスキーマ情報を定義する。

【００３６】ＡＰＩ１２は、サーバプログラム１３、オートメーションサーバ１４、或いはプログラム自動生成装置３からデータベース１１にアクセスするためのインタフェースである。

【００３７】サーバプログラム１３は、データベース１１に定義されているスキーマクラスに含まれるオブジェクトのメソッドにメッセージを送るためのメソッドの実装などの処理を行うためのプログラムである。

【００３８】オートメーションサーバ１４は、後述するようにプログラム自動生成装置３によって生成されるオブジェクト指向型のプログラムである（以下、オートメーションサーバ１４を構成するオブジェクトをオートメーションオブジェクトという）。オートメーションサーバ１１は、データベース１１のスキーマ情報を構成するクラスとその属性、メソッドを、オートメーションの機能によって外部のアプリケーションに公開する。リモートオートメーションマネージャ１５は、既存のプログラムによって実現され、インターネット５を介してオートメーションサーバ１４が公開するデータベース１１のスキーマを構成するクラス（以下、スキーマクラスという）とその属性、メソッドを、クライアント２に提供する。

【００３９】クライアント２は、プロセッサ、記憶装置及び入出力装置を有する情報処理装置によって構成され、クライアントプログラム２１とリモートオートメーションプロキシ２２の機能を含む。

【００４０】クライアントプログラム２１は、オートメーションサーバ１４が公開するデータベース１１のスキーマクラスとその属性、メソッドを利用する外部アプリケーションとなり、サーバ１のデータベース１１にアクセスする。リモートオートメーションプロキシ２２は、既存のプログラムによって実現され、クライアントプログラム２１がオートメーション通信でサーバ１が提供する機能を利用するために用いられる。

【００４１】プログラム自動生成装置３は、プロセッサ、記憶装置及び入出力装置を有する情報処理装置によって構成される。プログラム自動生成装置３は、データベース１１からＡＰＩ１２を介してそのデータベース１１に定義されているスキーマ情報を取得し、取得したスキーマ情報とプログラムテンプレートファイル４を基にして、オートメーション通信のためのオートメーションサーバ１４のためのプログラムを自動生成する。生成されたプログラムは、さらにコンパイルされて、オートメーションサーバ１４となる。

【００４２】プログラムテンプレートファイル４は、後述するオートメーションオブジェクトのプログラムを生成するためや、後述するＯＤＬ記述ファイルを生成するために用いられるテンプレートからなるファイルである。なお、プログラムテンプレートファイル４は、プログラム自動生成装置３が有する記憶装置内に記憶されていても、プログラム自動生成装置３の外部に設けられた記憶装置内に記憶されていてもよい。

【００４３】以下、図１に示すクライアント２とサーバ１とに別れたデータベース操作システムを構築する手順について、図２のフローチャートを参照して説明する。

【００４４】開発者は、まず、データベースマネジメントシステム（ＤＢＭＳ）を用いてデータベースのスキーマを定義し、データを登録することによってデータベース１１を構築する（ステップＳ１０１）。

【００４５】開発者は、次に、ＤＢＭＳが用意するＡＰＩ１２を用いてサーバプログラム１３を作成し、データベース１１のスキーマクラスを構成するオブジェクトにメッセージを送るためのメソッドを実装する（ステップＳ１０２）。

【００４６】以上のステップＳ１０１で構築されるデータベース１１、ステップＳ１０２で作成されるサーバプログラム１３は、既存のデータベース及びデータベースアプリケーションを利用するものとすることができる。

【００４７】次に、プログラム自動生成装置３は、データベースのスキーマクラスのオートメーションサーバ１４と、後述するクライアント２のセットアップ用に用いられるレジストリファイルとを自動生成する（ステップＳ１０３）。

【００４８】以後は、サーバ１の側とクライアント２の側とに、処理が分かれる。サーバ１の側では、最初に、オートメーションサーバ１４のレジストリ登録が行われる（ステップＳ１０４）。このレジストリ登録の詳細については、後述する。

【００４９】サーバ１の側では、次に、リモートオートメーションマネージャ１５を実装する処理が行われる（ステップＳ１０５）。このリモートオートメーションマネージャ１５には、既存のオートメーション通信用のプログラムを利用することができる。また、既にサーバ１に実装されているものを利用することもできる。リモートオートメーションマネージャの実装が終了すると、サーバ１の側におけるシステムの構築が終了する。

【００５０】クライアント２の側では、最初に、オートメーションサーバ１４のレジストリ登録が行われる（ステップＳ１０６）。このレジストリ登録の詳細について

は、後述する。

【００５１】クライアント２の側では、次に、リモートオートメーションプロキシ２２を実装する処理が行われる（ステップＳ１０７）。このリモートオートメーションプロキシ２２には、既存のオートメーション通信用のプログラムを利用することができる。また、既にクライアント２に実装されているものを利用することもできる。

【００５２】クライアント２の側では、次に、オートメーションの機能をサポートする開発言語を用いて、インターネット５を介してオートメーションサーバ１４を操作するための、クライアントプログラム２１が作成される（ステップＳ１０８）。クライアントプログラム２１が作成されると、クライアント２の側におけるシステムの構築の処理が終了する。

【００５３】以下、図１のプログラム自動生成装置３によるオートメーションサーバ１４及びレジストリファイルの生成の処理（ステップＳ１０３）の概要について、図３のフローチャートを参照して説明する。

【００５４】プログラム自動生成装置３は、最初に、開発者が入力装置を操作することによって、初期設定値を入力する（ステップＳ２０１）。

【００５５】ここで、入力する初期設定項目としては、例えば、①生成すべきオートメーションサーバ１３の名前、②スキーマ情報を読み込むデータベース１１の名前、③生成されたオートメーションサーバ１３で公開されるクラスとなるオートメーションオブジェクトの名前、④サーバ１との間の通信プロトコル種別、⑤サーバ１のネットワークアドレス、などがある。

【００５６】プログラム自動生成装置３は、次に、ＡＰＩ１２を介して登録されているクラス、クラスの属性、クラスのメソッドのそれぞれの名前と型の情報からなるデータベース１１のスキーマ情報を読み込む（ステップＳ２０２）。ステップＳ２０２の処理を終了すると、プログラム自動生成装置３は、次に、ステップＳ２０３、Ｓ２０４、Ｓ２０６の処理を並行して行う。

【００５７】ステップＳ２０３では、プログラム自動生成装置３は、プログラムテンプレートファイル４中のオートメーションオブジェクトに対応するテンプレートとなるファイルに、ステップＳ２０２で取得したスキーマ情報のコードを挿入することによって、オートメーションオブジェクトプログラムＤ２０１を生成する。

【００５８】ステップＳ２０４では、プログラム自動生成装置３は、プログラムテンプレートファイル４中のＯＤＬ記述ファイルのテンプレートとなるファイルに、ステップＳ２０２で取得したスキーマ情報のコードを挿入することによって、ＯＤＬ記述ファイルＤ２０２を生成する。

【００５９】ステップＳ２０３とＳ２０４の両方の処理が終了すると、プログラム自動生成装置３は、オートメーションオブジェクトプログラムＤ２０１と、ＯＤＬ記述ファイルＤ２０２と、サーバプログラムＤ２０３と、その他のオートメーションやサーバに必要なプログラムＤ２０４とを、それぞれプログラム開発言語環境に応じたコンパイラを用いてコンパイルし、さらにリンケージエディタを用いてリンクする（ステップＳ２０５）。これにより、オートメーションサーバ１４のプログラムが生成される。

【００６０】ここで、サーバプログラムＤ２０３は、図１のサーバプログラム１３に相当するものである。その他のオートメーションサーバに必要なプログラムＤ２０４は、開発者が用いるサーバ１やプログラム開発言語環境に特有のプログラムである。

【００６１】一方、ステップＳ２０６では、プログラム自動生成装置３は、クライアント２からサーバ２に接続するためのレジストリファイルＤ２０５を生成する。

【００６２】以上の処理によって、オートメーションサーバ１４のプログラムと、レジストリファイルＤ２０５とが生成されると、このフローチャートの処理を終了する。

【００６３】以下、図３のフローチャートにおける各ステップの処理について、詳細に説明する。

【００６４】まず、ステップＳ２０２のＡＰＩ１２からスキーマ情報を読み込む処理について説明する。ここでは、プログラム自動生成装置３は、データベース１１が稼働していることを条件として、データベース１１に登録されているスキーマクラスをすべて取得することができるＡＰＩやＡＰＩ群、或いはクラス毎にそのすべての属性の名前と型とアクセス識別子、さらにすべてのメソッドの名前と返却値の型と、すべての引数の型と、アクセス識別子とを取得することができるＡＰＩやＡＰＩ群をＡＰＩ１２として用いて、データベース１１からスキーマ情報を読み込む。

【００６５】次に、ステップＳ２０３のオートメーションプログラムオブジェクトのプログラムＤ２０１を生成する処理について説明する。ステップＳ２０３では、①プロパティ、②メソッド、③コンストラクタに分けて、オートメーションオブジェクトＤ２０１が生成される。

【００６６】①のプロパティを生成するとき、プロパティの値の取得と設定を実現しなければならないので、取得関数と設定関数とが用意される。以下、それぞれの関数の生成手順について、図４と図５のフローチャートを参照して説明する。

【００６７】なお、以下の説明において、Ｖａｒｉａｎｔ型とは、数値型、バイト型、文字列型、ブール型、日付型、通貨型、オブジェクト型などの数種類の型が定義されている共用体を持ち、その共用体で変数値の型を設定できる型のことを言う。

【００６８】プロパティの取得関数を生成する場合は、図４のフローチャートに示すように、プログラム自動生

成装置３は、まず、戻り値をＶａｒｉａｎｔ型で変数宣言し、その変数の型を上記のいずれかのうちの１つに設定する（ステップＳ３０１）。ここで、設定する変数の型は、ステップＳ２０２でＡＰＩ１２を介してデータベース１１から取得したスキーマ情報中のクラスの属性の型によって判断される。

【００６９】例えば、クラスの属性の型がｃｈａｒ＊型であったならば、変数の型は文字列型を表すＶＴ＿ＢＳＴＲに設定される。或いは、クラスの属性の型がｉｎｔ型であったならば、変数の型は、数値型のうちの整数型を表すＶＴ＿Ｉ２（或いは長整数型ＶＴ＿Ｉ４）に設定される。クラスの属性の型が他のクラスのオブジェクトを指すポインタ型であったならば、変数の型は、オブジェクト型を表すＶＴ＿ＤＩＳＰＡＴＣＨに設定される。

【００７０】プログラム自動生成装置３は、次に、ＤＢＭＳのＡＰＩ１２を介してデータベース１１にアクセスし、データベース１１に格納されている目的とするオブジェクトの属性値を取得する（ステップＳ３０２）。

【００７１】プログラム自動生成装置３は、次に、ステップＳ３０２で取得した属性値を、ステップＳ３０１で変数宣言した変数に設定した型で代入する。この代入の結果、返却された値がプロパティの取得関数のプロパティ値となる（ステップＳ３０３）。そして、プロパティの取得関数の生成の処理を終了する。

【００７２】プロパティの設定関数を生成する場合は、図５のフローチャートに示すように、プログラム自動生成装置３は、最初に、関数内で有効なローカル変数を、関数言語で指定される型、すなわち対象とする属性の型で宣言する（ステップＳ４０１）。

【００７３】プログラム自動生成装置３は、次に、ステップＳ４０１で指定されたＶａｒｉａｎｔ型の引数を取り出し、それをステップＳ４０１で宣言したローカル変数に代入する（ステップＳ４０２）。

【００７４】プログラム自動生成装置３は、次に、ＡＰＩ１２を介してデータベース１１にアクセスし、データベース１１に格納されている目的とするオブジェクトの属性値に、ローカル変数の値を代入（またはコピー）する（ステップＳ４０３）。

【００７５】プログラム自動生成装置３は、さらに、ＡＰＩ１２を介してデータベース１１にオブジェクトの属性を更新したことを通知し、これによってプロパティ値をデータベース１１に格納されている目的とするオブジェクトの属性に反映させて、更新させるためのプログラムとして必要となるコードを生成する（ステップＳ４０４）。そして、プロパティの設定関数の生成の処理を終了する。

【００７６】②のオートメーションオブジェクトのメソッドの生成手順について、図６のフローチャートを参照して、詳しく説明する。

【００７７】プログラム自動生成装置３は、まず、関数内で有効となるＶａｒｉａｎｔ型のローカル変数を宣言し、メソッドからの戻り値の型に対応するＶａｒｉａｎｔ型の型を設定する（ステップＳ５０１）。

【００７８】プログラム自動生成装置３は、次に、各引数毎に対応する開発言語仕様の型のローカル変数を用意し、その変数にそれぞれのＶａｒｉａｎｔ型の引数の値を代入する（ステップＳ５０２）。

【００７９】プログラム自動生成装置３は、次に、各引数が代入されたローカル変数の値をパラメータ値として、ＡＰＩ１２を介してデータベース１１に定義されているクラスのメソッドを実行させる（ステップＳ５０３）。

【００８０】プログラム自動生成装置３は、次に、ステップＳ５０３でのメソッドの実行の結果、値が変わったローカル変数の値を、対応する型でＶａｒｉａｎｔ型の引数の値に代入し、メソッドに実行による値の変化を反映させる（ステップＳ５０４）。

【００８１】プログラム自動生成装置３は、さらに、ステップＳ５０３でのメソッドの実行後に返却された値を、ステップＳ５０１で用意したＶａｒｉａｎｔ型のローカル変数に、対応する型で代入し、それをオートメーションメソッドの返却値とする（ステップＳ５０５）。そして、このフローチャートの処理を終了する。

【００８２】③のオートメーションオブジェクトのコンストラクタの生成手順について、詳しく説明する。

【００８３】オートメーションオブジェクトは、データベース１１のスキーマクラスを代理するものなので、プログラム自動生成装置３は、オートメーションオブジェクトのクラス定義に、そのスキーマクラスの参照型（ポインタ型）のオブジェクトを保持するメンバ変数の宣言を自動生成する。

【００８４】そして、プログラム自動生成装置３は、スキーマクラスの参照型のオブジェクトを生成し、生成したオブジェクトにメンバ変数を代入する文を挿入することによって、オートメーションオブジェクトのコンストラクタを生成する。

【００８５】オートメーションオブジェクトのコンストラクタは、クライアントプログラム２１からオートメーションオブジェクトのオブジェクト型変数の宣言があったときに呼び出される。従って、クライアント２の側で扱うオブジェクト型の変数は、スキーマクラスの参照型のオブジェクトということになる。

【００８６】もっとも、スキーマクラスの参照型は、データベース１１のオブジェクトを指すポインタであるので、オートメーションオブジェクトのコンストラクタが呼ばれるときに、スキーマクラスのオブジェクトが生成される訳ではない。つまり、オートメーションオブジェクトのコンストラクタとスキーマクラスのコンストラクタとは異なるものであり、スキーマクラスのオブジェクトの生成とそれに伴うコンストラクタの呼び出しは、別

にメンバ関数を用意することによって対応が図られている。

【００８７】次に、ステップＳ２０４のオートメーションサーバのＯＤＬ記述ファイルＤ２０２を生成する処理について、図７のフローチャートを参照して説明する。この処理は、図３のステップＳ２０２で、ＡＰＩ１２を介してデータベース１１から読み込んだスキーマクラスのすべての情報に対して適用される。

【００８８】プログラム自動生成装置３は、まず、ＣＯＭのＡＰＩである“CoCreateGuidを用いて、各クラス毎にディスパッチインタフェース用のＩＤを割り当て、ＯＤＬ記述ファイルに記述する（ステップＳ６０１）。

【００８９】プログラム自動生成装置３は、次に、ディスパッチインタフェースの名前としてスキーマクラスの名前に接頭語“Ｉ”を付け足した名前を用いて、ＯＤＬ記述ファイルの“dispinterface”文を記述する。“dispinterface”文では、“properties”と“methods”とを記述するが、ここでは、スキーマクラスの属性名とメソッドとがそのまま用いられる。また、返却値の型は、すべてＶａｒｉａｎｔ型で、引数の型はすべてＶａｒｉａｎｔのアドレスの型で宣言される。また、“properties”と“methods”とのＤＩＳＰＩＤには、１から通し番号が付けられる（ステップＳ６０２）。

【００９０】プログラム自動生成装置３は、次に、“CoCreateGuid”を用いて各クラス毎にオートメーションオブジェクト用のＩＤを割り当て、ＯＤＬ記述ファイルに記述する（ステップＳ６０３）。

【００９１】プログラム自動生成装置３は、さらに、“dispinterface”文と１対１で対応するように、m同じスキーマの名前で“coclass”文を記述する。“coclass”文の中には、その“coclass”の名前に接頭語“Ｉ”を付けた“dispinterface”文をサポートする宣言を記述する（ステップＳ６０４）。そして、このフローチャートの処理を終了する。

【００９２】次に、ステップＳ２０６のクライアントセットアップ用のレジストリファイルＤ２０５を生成する処理について、説明する。

【００９３】ステップＳ２０６で生成されるレジストリファイルＤ２０５は、例えば、Ｗｉｎｄｏｗｓ９５やＷｉｎｄｏｗｓＮＴ（いずれも登録商標）のレジストリエディタで、クライアン２のレジストリなどにオートメーションサーバ１４の情報を登録するために用いられる。

【００９４】レジストリに登録すべきオートメーションサーバ１４の情報には、オートメーションオブジェクト毎に、①オートメーションサーバの名前、②オートメーションオブジェクトの名前、③クラスＩＤ、④通信プロトコル種別、⑤ネットワークアドレス（ＩＰアドレス）、⑥クライアントプログラム２１の実行時に必要なダイナミックリンクライブラリなどが含まれる。

【００９５】このうち、①～⑤は、ステップＳ２０１で初期設定入力された情報と、ステップＳ２０２で読み出されたスキーマ情報とから取得される。⑥は、固定的なものであり、プログラムテンプレートファイル４中のレジストリファイルのテンプレートファイルに記述することによって用意される。

【００９６】プログラム自動生成装置３は、プログラムテンプレートファイル４中のレジストリファイルのテンプレートとなるファイルに、これらの情報のコードを挿入することによってレジストリファイルＤ２０５を生成する。

【００９７】以下、図３のフローチャートにおけるサーバ１の側でのオートメーションサーバ１４のレジストリ登録（ステップＳ１０５）、及びクライアント２の側でのオートメーションサーバ１４のレジストリ登録と接続設定（ステップＳ１０７）の手順について、詳しく説明する。

【００９８】ステップＳ１０５において、レジストリ登録される情報は、図２のステップＳ２０６で生成されたレジストリファイルＤ２０５に含まれる情報のうちの①オートメーションサーバの名前、②オートメーションオブジェクトの名前、③クラスＩＤである。これらの情報は、通常、オートメーションサーバ１４の起動時に実行されるレジストリ登録用のＡＰＩによってオートメーションサーバ１４のレジストリに登録することができる。

【００９９】ステップＳ１０７において、図２のステップＳ２０６で生成されたレジストリファイルＤ２０５に含まれる情報が、例えば、Ｗｉｎｄｏｗｓ９５やＷｉｎｄｏｗｓＮＴのレジストリエディタに読み込ませて、クライアントプログラム２１のレジストリに登録される。なお、通信プロトコル種別やネットワークアドレスなどのサーバ１への接続に関する設定は、レジストリファイルＤ２０５をクライアント２１のレジストリに登録した後でも、例えば、リモートオートメーション接続マネージャなどを用いて変更することが可能である。

【０１００】以下、上記のようにして構築したシステムで実行されるデータベースアプリケーションプログラムで、オブジェクトの多相性が如何にして実現されているかについて、詳しく説明する。

【０１０１】この実施の形態のシステムにおいて、多相性プログラムの実現方法は、オートメーションオブジェクトに多相性を実現すべきオブジェクトのクラス名を記録させる属性を追加し、プログラムの実行時には、オートメーションオブジェクトは、その属性から得たクラス名によって当該オブジェクトのメソッドを呼び出すという考え方に基づいている。

【０１０２】また、オートメーションサーバ１４が公開するオートメーションオブジェクトの引数の型をすべてＶａｒｉａｎｔ型とし、そのＶａｒｉａｎｔ型の変数値の型をオートメーションサーバ１４が判別することによって、適切なメソッドを呼び出している。以下、メソッ

ドの多重定義の場合と、仮想関数との場合に分けて、多相性の実現方法を詳しく説明する。

【０１０３】オブジェクトの多相性を実現するために、オートメーションオブジェクトは、引数の型及び／または個数が異なるメソッドを複数定義している。ここで、メソッドの引数の最小個数をｍ（自然数）、最大個数をＭ（自然数）とすると、ＯＤＬ記述ファイルは、①Ｖａｒｉａｎｔ型の戻り値、②メソッド名、③Ｍ個のＶａｒｉａｎｔ型の引数、④Ｍ－ｍ個のｏｐｔｉｏｎａｌパラメータを指定したＶａｒｉａｎｔ型の引数によってメソッドを定義する。

【０１０４】④のｏｐｔｉｏｎａｌパラメータは、オートメーションメソッドの呼び出し時に、パラメータによって対応する引数の値を省略すべきことを示すものである。但し、ｏｐｔｉｏｎａｌパラメータによる指定は、すべてＶａｒｉａｎｔ型の引数でなければならないという制限がある。

【０１０５】次に、仮想関数の場合、仮想関数は、サーバ１のが側の開発言語や、データベース１１にアクセスするためのＡＰＩ１２がサポートしているので、仲介者となるオートメーションサーバ１４は、クライアントプログラム２１がどのオブジェクトのメソッドを呼び出したのかをサーバプログラム１３に渡せばよいこととなる。

【０１０６】仮想関数の処理においては、“virtual”で定義しているメンバ変数をもつ親クラス（スーパークラス）のオブジェクトと、その親クラスから派生したクラス（派生クラス：サブクラス）のオブジェクト間の実行時の切換が重要となる。

【０１０７】仮想関数の運用において、まず、親クラスのオブジェクトへのポインタを宣言し、そのポインタに対してメンバ関数の呼び出しが記述される。そして、実行時に、そのポインタが指すオブジェクトの実体（インスタンス）が、親クラスのオブジェクトであるか派生クラスのオブジェクトかをシステムが判断し、そのどちらか一方の正しいオブジェクトのメソッドが呼び出される。

【０１０８】この実施の形態のシステムにおいては、クライアント２の側で用いられるオートメーションオブジェクトは、オートメーションサーバ１４においてスキーマクラスのオブジェクトを指すポインタとして実装されている。従って、オートメーションサーバ１４が仮想関数の処理を正しく中継するためには、クライアント２の側で親クラスのオブジェクト型変数の実体が実行時に派生クラスのオブジェクト型変数に置き換わったときに、オートメーションサーバ１４も同様のポインタの置き換えを行えばよい。

【０１０９】この実施の形態のシステムでは、このポインタの置き換えについて、ポインタ型の代入演算子の代わりとなるメンバ関数を親クラスのオートメーションオブジェクトに用意することによって、対応している。このメンバ関数は、派生クラスのオブジェクト変数を引数にとり、そのメンバ関数が呼ばれる親クラスのオブジェクトへのポインタを、引数である派生クラスのオブジェクトのポインタに置き換える処理を行うものである。

【０１１０】この処理によって、クライアントプログラム２１で親クラスのオブジェクトからのメンバ変数の呼び出しが記述されていても、そのオブジェクトの実体を実行時に置き換えることが可能となり、オートメーションサーバ１４において、仮想関数の振る舞いを実現することが可能となる。

【０１１１】以下、上記のようにして構築したシステムにおいて、クライアントプログラム２１からデータベース１１へアクセスするための処理について、図８のシーケンス図を参照して説明する。

【０１１２】なお、ここで、クライアントプログラム２１とオートメーションサーバ１４との間の通信は、リモートオートメーションプロキシ２２とリモートオートメーションマネージャ１５とを用い、インターネット５を介して行われる。また、オートメーションサーバ１４或いはサーバプログラム１３とデータベース１１との間のデータのやり取りは、ＡＰＩ１２を介して行われる。

【０１１３】以下、図８のシーケンス図に示す処理において、オートメーションサーバ１４が実行する処理の詳細について、図９のフローチャートを参照して説明する。

【０１１４】オートメーションサーバ１４は、まず、クライアントプログラム２１から呼び出されたメソッドの引数の個数をチェックし、引数が１つ以上あるかどうかを判定する（ステップＳ７０１）。

【０１１５】ステップＳ７０１で引数が１つ以上ない、すなわち引数がないと判定されたときは、オートメーションサーバ１４は、引数なしのメソッドを呼び出す（ステップＳ７０２）。そして、このフローチャートの処理を終了する。

【０１１６】ステップＳ７１１で引数が１つ以上あると判定されたときは、オートメーションサーバは、メソッドの第１引数の値からすべての引数の値の処理が終了するまで、ステップＳ７０３－Ｓ７０３’のループの処理を繰り返す。もっとも、後述するように、すべての引数の値の処理を終了する前に、このループから抜け出す場合はある。

【０１１７】ステップＳ７０３－Ｓ７０３’のループの処理では、オートメーションサーバ１４は、まず、メソッドの引数がｏｐｔｉｏｎａｌ指定されているかどうかを判定する（ステップＳ７０４）。

【０１１８】ステップＳ７０４でメソッドの引数がｏｐｔｉｏｎａｌ指定されていると判定されたときは、オートメーションサーバ１４は、ｏｐｔｉｏｎａｌ指定の値が設定されているかどうかを判定する（ステップＳ７０

５）。

【０１１９】ステップＳ７０５でｏｐｔｉｏｎａｌ指定の値が設定されていないと判定されたときは、このループを抜けて、後述するステップＳ７１１に進む。ステップＳ７０５でｏｐｔｉｏｎａｌ指定の値が設定されていると判定されたときは、ステップＳ７０６に進む。また、ステップＳ７０４でメソッドの引数がｏｐｔｉｏｎａｌ指定されていないと判定されたときも、後述するステップＳ７０６に進む。

【０１２０】ステップＳ７０６では、オートメーションサーバ１４は、Ｖａｒｉａｎｔ型引数の値がどのような型であるかをチェックする。オートメーションサーバ１４は、さらにステップＳ７０６でチェックした引数の型がオブジェクト型であるかどうかを判定する（ステップＳ７０７）。

【０１２１】ステップＳ７０７で引数の型がオブジェクト型であると判定されたときは、オートメーションサーバ１４は、Ｖａｒｉａｎｔ型のオブジェクト型の変数のクラス名を、各クラスで実現されているオブジェクト自身にそのオブジェクトのクラス名を記録させておくリフレクションの機能を用いて、チェックする（ステップＳ７０８）。

【０１２２】オートメーションサーバ１４は、さらに、クラス名に対応するスキーマクラスのオブジェクトを宣言し、宣言したオブジェクトに、ステップＳ７０７で判定されたオブジェクト型の引数を代入する（ステップＳ７０９）。ステップＳ７０９の処理が終了すると、メソッドに次の引数があればその引数についてのループ（Ｓ７０３－Ｓ７０３’）の処理に、次の引数がなければ、ループ（Ｓ７０３－Ｓ７０３’）を抜けて、ステップＳ７１１の処理に進む。

【０１２３】ステップＳ７０７で引数の型がオブジェクト型でないと判定されたときは、オートメーションサーバ１４は、ステップＳ７０６でチェックした型に対応する開発言語仕様の型で変数宣言をし、宣言した変数に引数の値を代入する（ステップＳ７１０）。ステップＳ７１０の処理が終了すると、メソッドに次の引数があればその引数についてのループ（Ｓ７０３－Ｓ７０３’）の処理に、次の引数がなければ、ループ（Ｓ７０３－Ｓ７０３’）を抜けて、ステップＳ７１１の処理に進む。

【０１２４】ステップＳ７１１では、オートメーションサーバ１４は、以上の処理で準備された変数やオブジェクトを引数として、多重定義されたメソッドを呼び出す。そして、このフローチャートの処理を終了する。

【０１２５】以上説明したように、プログラム自動生成装置３は、データベース１１に定義されているスキーマクラスの情報を取得し、取得した情報のうちの適切なものをテンプレートファイル４のうちのオートメーションサーバ生成用のものに挿入することによって、オートメーションサーバ１４を生成することができる。また、取得した情報のうちの適切なものをテンプレートファイル４のうちのＯＤＬ記述ファイル生成用のものに挿入することによって、ＯＤＬ記述ファイルを生成し、これにより、インタフェース定義を定めることができる。さらに、取得した情報からレジストリ登録に使用する情報からなるレジストリファイルも生成することができる。

【０１２６】従って、この実施の形態のプログラム自動生成装置３によれば、オブジェクト指向のデータベース１１を操作するためのプログラムの生成が容易になる。また、例えば、インタフェース仕様が変更された場合などにも、プログラムを一から作り直す必要がないので、プログラムの運用、保守も容易になる。

【０１２７】しかも、データベース１１、クライアントプログラム２１、或いはオートメーション通信を行うためのリモートオートメーションマネージャ１５及びリモートオートメーションプロキシ２２には、既存のシステムを利用することができるので、プログラムの開発、さらにはテストを容易に行うことができる。

【０１２８】さらに、ｏｐｔｉｏｎａｌパラメータの導入によって、クライアント２側のクライアントプログラム２１とサーバ１側のオートメーションサーバ１４とのオートメーション通信におけるインタフェース仕様を統一することができ、なおかつデータベース１１に登録されているオブジェクトの多相性の機能を十分に利用することができる。

【０１２９】上記の実施の形態では、プログラム自動生成装置３は、サーバ１とは別個の情報処理装置によって構成されていた。しかしながら、サーバ１がプログラム自動生成装置３の機能を含むものとして、サーバ１とプログラム自動生成装置３とを同一の情報処理装置によって構成することもできる。この場合において、プログラムテンプレートファイル４は、サーバ１内の記憶装置に記憶することも外部の記憶装置に記憶することもできる。

【０１３０】上記の実施の形態では、オートメーションオブジェクトプログラム、ＯＤＬ記述及びレジストリファイルは、すべて同一のプログラム自動生成装置３によって生成されていた。しかしながら、これらは、それぞれ別々の情報処理装置によって生成し、コンパイルした後に、リンクしてもよい。

【０１３１】上記の実施の形態では、インターネット５を介して接続されたサーバ１とクライアント２とからなるクライアントサーバシステムで本発明のプログラムの開発・実行環境を実現した場合について説明した。しかしながら、本発明はこれに限られない。例えば、イントラネット環境でのクライアントサーバシステムを本発明のプログラムの開発・実行環境とすることもできる。

【０１３２】また、クライアント（マシン）２とサーバ（マシン）１とが別個に存在するクライアントサーバシステムではなく、図１０に示すようなスタンドアローン

型のコンピュータシステム６も、本発明のプログラムの開発・実行環境とすることができる。この場合、クライアントプログラム２１のプロセスとコンピュータシステム６上にオートメーションサーバ１４のサーバプロセスとが存在することとなり、クライアントプログラム２１とオートメーションサーバ１４とがプロセス間通信の手法を用いてオートメーション通信を行う。

【０１３３】上記の実施の形態では、ＣＯＭ／ＤＣＯＭを基礎とするオートメーション通信を例として説明した。しかしながら、本発明におけるオートメーション通信は、ＣＯＭ／ＤＣＯＭを基礎とするものに限るものではなく、本発明の趣旨を逸脱しない限りにおいて、他の仕様を基礎とするものにも適用することができる。

【０１３４】

【実施例】以下、上記の実施の形態のデータベースアプリケーションプログラムの開発・実行環境を具体的なシステムで実現した場合の実施例について、プログラムリスト（ソースリスト）を参照しながら説明する。

【０１３５】この実施例のシステムでは、データベース１１としてオブジェクト指向データベースＰＥＲＣＩＯ（登録商標）、サーバプログラム開発言語としてＶｉｓｕａｌ　Ｃ＋＋（登録商標）、クライアントプログラム開発言語としてＶｉｓｕａｌＢａｓｉｃ（登録商標）を用いている。

【０１３６】１）オートメーションオブジェクトのクラス

オートメーションオブジェクトのヘッダファイルの一部となるプログラムテンプレートファイル４のプログラムリストの一例を以下に示す。この例では、“@@”で囲まれる文字列が、オートメーションサーバ１４がＡＰＩを介してデータベース１１から取得するスキーマクラスの情報が埋め込まれる個所である。従って、以下の“@@CLASS@@”は、クラス名に置き換えられる。

【０１３７】また、“dRef”は、すべてのユーザ定義スキーマクラスの基本クラスとなっている。オートメーションオブジェクトのクラスは、“dRef”を継承して定義されている。“d_Extend”“d_Set”、“d_List”は、ＰＥＲＣＩＯがシステムとして予め用意しているテンプレート型のスキーマクラスである。“d_Ref_Any”は、ＰＥＲＣＩＯがシステムとして予め用意しているデータベース中の任意のオブジェクトへのポインタを保持する変数の型である。

【０１３８】オートメーションオブジェクトのクラスは、これらのシステムが用意するスキーマクラスのオブジェクトを生成するメンバ関数“createextend()”、“createset()”、“createlist()”の関数宣言と、自分自身を指すポインタを生成するメンバ関数“create_newref()”の関数宣言とを定義する。

【０１３９】なお、この実施例で示されている、接頭語“C”で始まるクラスは、オートメーションオブジェクトのクラスを表し、接頭語“C”をとったクラス名のスキーマクラスと１対１で対応する。また、接頭語“d_”で始まるクラスは、ＰＥＲＣＩＯがシステムとして予め用意するスキーマクラスであり、アンダースコアのない接頭語“d”で始まるクラスは、ＰＥＲＣＩＯがシステムとして用意するクラスと１対１で対応するオートメーションクラスを表している。

【０１４０】

```
class C@@CLASS@@ : public dRef
{
  DECLARE_DYNCREATE(C@@CLASS@@)
  C@@CLASS@@();
public:
  d_Extent<d_Ref_Any>* createextent();
  d_Set<d_Ref_Any>* createset();
  d_List<d_Ref_Any>* createlist();
  LPDISPATCH create_newref();
```

【０１４１】２）オートメーションオブジェクトの基本クラスのプログラム

オートメーションオブジェクトのクラス“dRef”のヘッダファイルのプログラムリストの一部を以下に示す。“dRef”は、すべてのユーザ定義スキーマクラスに共通の基本クラスであり、固定的なファイルとしてプログラムテンプレートファイル４に用意されている。

【０１４２】

```
class dRef : public CCmdTarget
{
  DECLARE_DYNCREATE(dRef)
  dRef();
public:
  d_Ref_Any m_classref;
  CString m_classname;
public:
  virtual d_Extent<d_Ref_Any>* createextent();
  virtual d_Set<d_Ref_Any>* createset();
  virtual d_List<d_Ref_Any>* createlist();
  virtual LPDISPATCH create_newref();
protected:
  virtual   dRef();
};
```

【０１４３】オートメーションオブジェクトのテンプレートファイルで定義されているメンバ変数は、すべて“dRef”クラスで、仮想関数で定義されている。また、属性の定義としては、オートメーションオブジェクトが代理として振る舞っているＰＥＲＣＩＯのオブジェクトへのポインタを保持する“m_classref”、オートメーションオブジェクト自身のクラス名を保持する“m_classname”がある。“m_classref”は、オートメーションオブジェクトのコンストラクタの説明の部分で述べたスキーマクラスの参照型のオブジェクトを保持するメンバ変

数である。また、“m_classname”は、多重定義の説明の部分で述べたオブジェクトのクラス名を記憶させるメンバ変数である。

【０１４４】３）オートメーションオブジェクトのプロパティのプログラム

図４のプロパティの取得関数の生成手順を実現するプログラムリストを以下に示す。このプログラムは、スキーマクラスのｉｎｔ型の属性値を取得するプロパティの取得関数を示すものである。この例では、“@@ORIGVAR@@”がスキーマクラスの情報の属性名と置き換わる。また、“@@VAR@@”は、その属性の中でアンダースコアを取り除いた文字列と置き換わる。

【０１４５】

```
VARIANT C@@CLASS@@::_Get@@VAR@@()
{
  VARIANT vaResult;
  VariantInit(&vaResult);
  V_VT(&vaResult) = VT_I4;
  if (!m_classref)
   return vaResult;
  V_I4(&vaResult) = GETOBJ(m_classref)->@@ORIGVAR@@;
  return vaResult;
}
```

【０１４６】図５のプロパティの設定関数の生成手順を実現するプログラムリストを以下に示す。このプログラムは、スキーマクラスのｃｈａｒ＊型の属性値を取得するプロパティの取得関数を示すものである。ここで、最後の行にある“GETOBJ”はマクロであり、“m_classref”のオブジェクトの参照型に展開される。また、“mark_modified()”は、ＰＥＲＣＩＯが用意するシステムのメンバ関数であり、プロパティ設定においてオブジェクトの属性値が変更されたことをデータベース１１に通知して、データベース１１の内容を更新する。

【０１４７】

```
void C@@CLASS@@::_Set@@VAR@@(const VARIANT FAR& newValue)
{
  CString c_@@VAR@@;
  if (!m_classref)
   return;
  c_@@VAR@@ = CString(V_BSTR(&newValue));
  strcpy(GETOBJ(m_classref)->@@ORIGVAR@@, c_@@VAR@@);
  GETOBJ(m_classref).mark_modified();
}
```

【０１４８】４）オートメーションオブジェクトのメソッドのプログラム

図６のオートメーションメソッドの生成手順を実現するプログラムリストを以下に示す。このプログラムは、ユーザが定義したスキーマクラス“Employee”を代理するオートメーションオブジェクトのクラス“CEmployee”のメソッド“getdata”を実装するものである。スキーマクラスのメソッド名は、“get_data”である。また、“CopyToVARIANT”は、グローバル関数であり、ＡＰＩ呼び出しの結果置き換わったパラメータ“n_param0”の値を、オートメーションメソッドのＶａｒｉａｎｔ型の引数“param0”にコピーする。

【０１４９】

```
VARIANT CEmployee::getdata(VARIANT FAR* param0)
{
  VARIANT vaResult;
  VariantInit(&vaResult);
  char *n_param0 = NULL;
  if (AllocAndCopyCharParam(&n_param0, param0)) {
    return vaResult;
  }
  GETOBJ(m_classref)->get_data(n_param0);
  CopyToVARIANT(n_param0, param0);
  if (n_param0)
    free(n_param0);
  return vaResult;
}
```

【０１５０】５）サーバプログラム

スキーマクラス“Employee”のメンバ関数get_dataの実装例を以下に示す。このプログラムは、図１のサーバプログラム１３に対応し、また、このプログラムの作成

は、図２のステップＳ１０２の処理に相当する。

【０１５１】

```
void Employee::get_data(char* data)
{
  char tempstr[1024];
  sprintf(tempstr, "%s¥t%s¥t%d¥t%s¥t%d¥r¥n",
    (char*)name,
    (char*)birth_day,
    get_age(),
    (char*)section,
    salary);
  strcpy(data, tempstr);
}
```

【０１５２】６）ＯＤＬ記述ファイル

ＯＤＬ記述ファイルを生成するためのプログラムテンプレートファイル４のうちのテンプレートの例を以下に示す。ここでは、“dispinterface”文の“propeties”までを記述するテンプレートファイルの例を示す。以下の例は、図７のＯＤＬ記述の生成手順において用いられる。

【０１５３】

```
dispinterface I@@CLASS@@
{
  properties:
    //{{AFX_ODL_PROP(C@@CLASS@@)
```

【０１５４】この後に、スキーマクラスの属性が挿入される。“dispinterface”文の“method”までを記述するテンプレートファイルを以下に示す。

【０１５５】

```
    //}}AFX_ODL_PROP
  methods:
    //{{AFX_ODL_METHOD(C@@CLASS@@)
```

【０１５６】この後に、スキーマクラスのメソッドが挿入される。“dispinterface”文の最後の部分の生成に用いられるテンプレートファイルを以下に示す。

【０１５７】

```
    //}}AFX_ODL_METHOD
};
// Class information for C@@CLASS@@
```

【０１５８】“coclass”文を記述するためのテンプレートファイルを以下に示す。

【０１５９】

```
coclass @@CLASS@@
{
  [default] dispinterface I@@CLASS@@;
};
```

【０１６０】上記の図７のフローチャートで示した手順に従ってこの実施例のプログラム生成装置３が生成した、オートメーションオブジェクトのクラス“CEmployee”のＯＤＬ記述ファイルの例を以下に示す。このＯＤＬ記述ファイルをコンパイルし、タイプライブラリを作成して公開すれば、オートメーションオブジェクトを外部と連携することが可能となる。

【０１６１】

```
[ uuid(5794F9E6-3A54-11d1-BB17-00004C95A764) ]
dispinterface IEmployee
{
  properties:
    //{{AFX_ODL_PROP(CEmployee)
    [id(1)] VARIANT name;
    [id(2)] VARIANT birthday;
    [id(3)] VARIANT section;
    [id(4)] VARIANT salary;
    //}}AFX_ODL_PROP
  methods:
    //{{AFX_ODL_METHOD(CEmployee)
    [id(5)] VARIANT create(VARIANT * db_obj, VARIANT* param0, VARIANT*
param1, VARIANT* param2, VARIANT*param3);
    [id(6)] VARIANT setsalary(VARIANT* param0);
    [id(7)] VARIANT getsalary();
    [id(8)] VARIANT getdata(VARIANT* param0);
    [id(9)] VARIANT getall(VARIANT* param0);
    [id(10)] VARIANT remove();
    //}}AFX_ODL_METHOD
```

```
};
// Class information for CEmployee
[ uuid(5794F9E7-3A54-11d1-BB17-00004C95A764) ]
coclass CEmployee
{
  [default] dispinterface IEmployee;
}
```

【０１６２】７）クライアントセットアップ用レジストリファイル

図３のステップＳ２０６で生成されるレジストリファイルＤ２０５の、この実施例における生成例を以下に示す。この例では、オートメーションサーバの名前が、“BookAX”、オートメーションオブジェクトの名前が“CEmployee”、オートメーションサーバ“BookAX”のＩＰアドレスが“133.207.62.173”と設定されている。また、オートメーションオブジェクト“CEployee”のＣＬＳＩＤは、“5794F9E7-3A54-11d1-BB17-00004C95A764”に設定されており、ＯＤＬ記述ファイル中の“coclass”文の“CEmployee”の“uuid”と一致する。なお、このレジストリファイルの内容は、図２のステップＳ１０６でクライアント２に登録される。

【０１６３】

```
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}]
@="BookAX.CEmployee"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥LocalServer32]
@="BookAX.EXE"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥_LocalServer32]
@="BookAX.EXE"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥ProgId]
@="BookAX.CEmployee"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥InprocHandler32]
@="ole32.dll"
[HKEY_CLASSES_ROOT¥BookAX.CEmployee]
@="BookAX.CEmployee"
[HKEY_CLASSES_ROOT¥BookAX.CEmployee¥CLSID]
@="{5794F9E7-3A54-11d1-BB17-00004C95A764}"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥InprocServer32]
@="autprx32.dll"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥NetworkAddress]
@="133.207.62.173"
[HKEY_CLASSES_ROOT¥CLSID¥{5794F9E7-3A54-11d1-BB17-00004C95A764}¥ProtocolSequence]
@="ncacn_ip_tcp"
```

【０１６４】８）クライアントプログラム

Ｖｉｓｕａｌ　Ｂａｓｉｃによって生成した、この実施例のクライアントプログラム２１の例を以下に示す。これは、図２のステップＳ１０８で生成されるものである。この例では、オートメーションサーバ１４には、“BookAX”と“CPPAPI”という２つの名前が用いられており、ユーザが定義したスキーマクラスを代理するオートメーションサーバには“BookAX”が、ＰＥＲＣＩＯが用意するシステムのスキーマクラスを代理するオートメーションサーバには“CPPAPI”が用いられている。

【０１６５】“Dim”ステートメントでＶａｒｉａｎｔ型の変数を宣言し、それに続く“CreateObject”関数でオートメーションオブジェとを参照し、“Set”ステートメントでＶａｒｉａｎｔ型の変数にオブジェクト型として保持させる。この２つのＶａｒｉａｎｔ型の変数によって、ＰＥＲＣＩＯのスキーマクラスのオブジェクトへの参照関係も解除される。

【０１６６】また、プログラムの後半で用いられている“Grid1”や“Text1”などで表されている“ActiveX”コントロールは、グリッドコントロールとテキストボックスコントロールとを用いて、データベース１１の検索結果を表示するときに用いられている。

【０１６７】

```
Private Sub printemp()
Dim emp As Variant
Set emp = CreateObject("BookAX.CEmployee")
Dim ext As Variant
Set ext = CreateObject("CPPAPI.dExtent")
ext.init emp
Dim itr As Variant
Set itr = ext.createiterator
Dim str As Variant
Dim sum As Variant
Dim i As Integer
Do While itr.Next(emp)
str = String(64, " ")
emp.GetData str
sum = sum + str
i = i + 1
Set str = Nothing
Loop
Grid1.Clip = sum
Text1.Text = i
```

```
Set emp = Nothing
Set ext = Nothing
Set itr = Nothing
End Sub
```

【０１６８】９）多重定義

スキーマクラスのメソッド“get_data”は、引数が最小で１個、最大で３個の多重定義されているメソッドであるとする。この場合、ＯＤＬ記述ファイルには、ｏｐｔｉｏｎａｌ指定を用いてオートメーションメソッド“getdata”を記述することとなるが、この実施例において、プログラムは以下のように記述される。

【０１６９】以下のようなｏｐｔｉｏｎａｌ指定をＯＤＬ記述ファイルに記述することによって、オートメーションサーバ１４は、クライアントプログラム２１から渡された引数の型をＶａｒｉａｎｔ型からデータベース１の開発言語の型のものに変換すれば、引数が１個から３個のいずれの個数のメソッドも呼び出すことができる。

```
[id(8)] VARIANT getdata(VARIANT* param0, [optional] VARIANT* param1, [optional] VARIANT* param2);
```

【０１７０】

【０１７１】１０）仮想関数

この実施例において、仮想関数を実現するための仮想関数の例を以下に示す。ここで、“CPerson”というクラスが、“CEmployee”の親クラスとなる。また、以下のプログラム中の“d_Ref<Person>”は、スキーマクラス<Person>のオブジェクトを指すポインタを保持する変数の型である。

【０１７２】

```
VARIANT CPerson::cast(VARIANT FAR* param0)
{
  dRef *src_dRef =
    dynamic_cast<dRef*>(CCmdTarget::FromIDispatch(V_DISPATCH(param0)));
  if(src_dRef != NULL) {
    d_Ref<Person> c_person;
    c_person = (d_Ref<Person>)src_dRef->m_classref;
    m_classref = (d_Ref_Any)c_person;
  }
}
```

【０１７３】以下、この実施例のシステムにおけるデータベースアプリケーションプログラムの開発プロセスについて、図１１のフローチャートを参照して説明する。

【０１７４】まず、開発者は、ＰＥＲＣＩＯ（データベース１１）のスキーマを設計し、データベースの構築を行う（ステップＳ８０１）。設計したデータベースのスキーマは、拡張及び／または変更をすることが可能である（ステップＳ８０２）。さらに、開発者は、各スキーマクラス毎にメソッドを実装する（ステップＳ８０３）。ここで実装したメソッドが、上記のサーバプログラム１３の一部となる。なお、以上のステップＳ８０３までで示したシステムの構築を行う代わりに、既存のシステムを用いることも可能である。

【０１７５】以後、開発プロセスは、システムの開発がサーバ１側とクライアント２側とで並行して進めることが可能となる。サーバ１の側では、後述するステップＳ８０４～Ｓ８１０のすべての開発プロセスを実行することが可能となる。一方、クライアント２の側では、後述するステップＳ８０５とＳ８０６、及びＳ８０８～Ｓ８１０までの開発プロセスを実行することが可能となる。

【０１７６】次に、開発者は、上記の実施の形態で示したプログラム自動生成装置３を用いて、データベース１１に登録されているスキーマクラスに対応したオートメーションサーバ１４を生成する（ステップＳ８０４）。

【０１７７】次に、開発者は、サーバ１の側のＰＥＲＣＩＯの環境を用いて、或いはオートメーションサーバ１４を介してクライアント２の側から操作することによって、データベース１１に実体（インスタンス）を格納する（ステップＳ８０５）。なお、いずれの方法による場合でも、データベース１１に格納したインスタンスは、Ｖｉｓｕａｌ　Ｂａｓｉｃなどのスクリプト的なインタプリタ言語の実行によって、データベース１１のデータやメソッドなどの振る舞いを確認することができる（ステップＳ８０６）。

【０１７８】次に、開発者は、ＰＥＲＣＩＯのＡＰＩを直接用いて、上記のプログラム自動生成装置３が生成するオートメーションサーバ１４と異なる独自のオートメーションサーバを生成して、両方のオートメーションサーバを併用するデータベースアプリケーションを構築する。

【０１７９】次に、開発者は、クライアント２をコントロールするための、“ActiveX”コントロールを実装する（ステップＳ８０８）。なお、このシステムはオートメーション通信の機能を用いた環境として構築されているため、クライアント２の側で動作させる“ActiveX”コントロールをアプリケーションの部品として用いることができる。また、アプリケーションに有用な既存の“ActiveX”コントロールが存在すれば、その再利用も可能である。

【０１８０】次に、上記のステップで開発した、プログ

ラム自動生成装置３を用いて作成したオートメーションサーバ１４、ＰＥＲＣＩＯのＡＰＩを用いて生成したオートメーションサーバ、及び“ActiveX”コントロールをクライアントプログラム２１の中で統合させる。ここでは、さらにＶｉｓｕａｌ　Ｂａｓｉｃなどのスクリプト的なインタプリタ言語を用い、アプリケーションのテスト実行、ＧＵＩのプロトタイプの作成などを行う（ステップＳ８０９）。

【０１８１】ステップＳ８０９までで、この実施例のシステムのためのデータベースアプリケーションは一応完成するが、オートメーション通信のためのプログラムの部分は、Ｖｉｓｕａｌ　Ｃ＋＋などのコンパイラ言語でも開発可能なので、開発者は、このようなコンパイラ言語を用いて、テストやプロトタイプでない最終的なデータベースアプリケーションを構築する（ステップＳ８１０）。

【０１８２】

【発明の効果】以上説明したように、本発明によれば、オブジェクト指向データベースを操作するためのプログラムを容易に作成することができる。また、そのプログラムの保守も容易になる。また、外部操作情報の生成によって、クライアントサーバ型のシステム構成をした場合のクライアント側のプログラムの作成も容易になる。

【０１８３】また、本発明は、オートメーション通信の機能を利用することによって、既存のシステムの利用が可能となり、プログラムの開発、運用及び保守が容易になる。

【０１８４】さらに、本発明は、インターフェース仕様を統一することによって、データベースに定義されているスキーマを構成するオブジェクトが多相性をサポートしている場合に、このオブジェクトの多相性の機能を十分に利用することができる。

【図面の簡単な説明】

【図１】本発明の実施の形態にかかるデータベースアプリケーションプログラムの開発・実行環境を実現するためのシステム構成例を示す図である。

【図２】図１に示すシステムを構築する手順を示すフローチャートである。

【図３】図１のプログラム自動生成装置によるプログラムの生成の処理の概要を示すフローチャートである。

【図４】プロパティの取得関数の生成手順を示すフローチャートである。

【図５】プロパティの設定関数の生成手順を示すフローチャートである。

【図６】オートメーションオブジェクトのメソッドの生成手順を示すフローチャートである。

【図７】オートメーションオブジェクトのＯＤＬ記述ファイルの生成手順を示すフローチャートである。

【図８】図１のクライアントプログラムからデータベースへアクセスするための処理の流れを説明する図である。

【図９】図１のオートメーションサーバによるメソッドの呼び出し手順を示す図である。

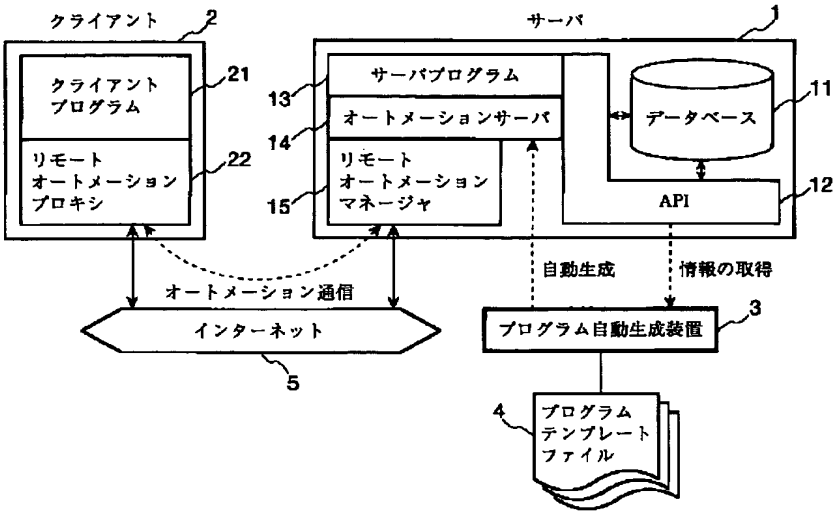
【図１０】本発明の実施の形態の具体例にかかるクライアントサーバ型のデータベース操作システムの開発プロセスを示すフローチャートである。

【図１１】本発明の他の実施の形態にかかるデータベースアプリケーションプログラムの開発・実行環境を実現するためのシステム構成例を示す図である。

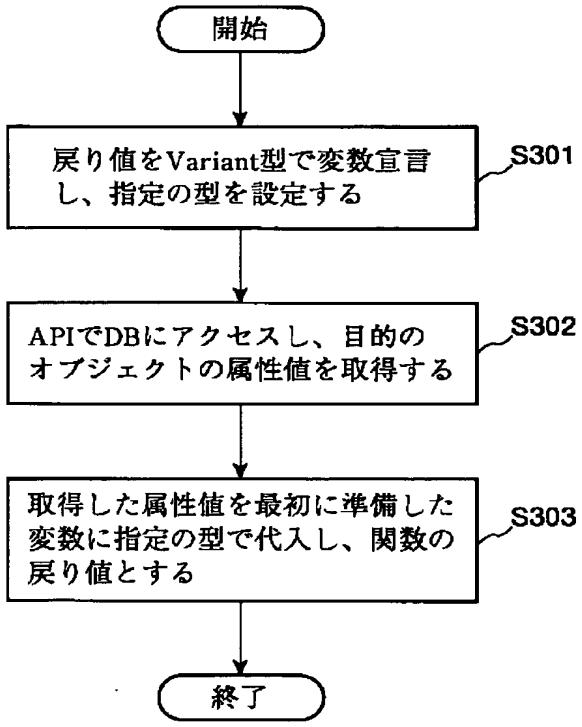
【符号の説明】

１　サーバ
２　クライアント
３　プログラム自動生成装置
４　プログラムテンプレートファイル
５　インターネット
６　スタンドアローン型のコンピュータシステム
１１　データベース
１２　ＡＰＩ（Application Programing Interface）
１３　サーバプログラム
１４　オートメーションサーバ
１５　リモートオートメーションマネージャ
２１　クライアントプログラム
２２　リモートオートメーションプロキシ

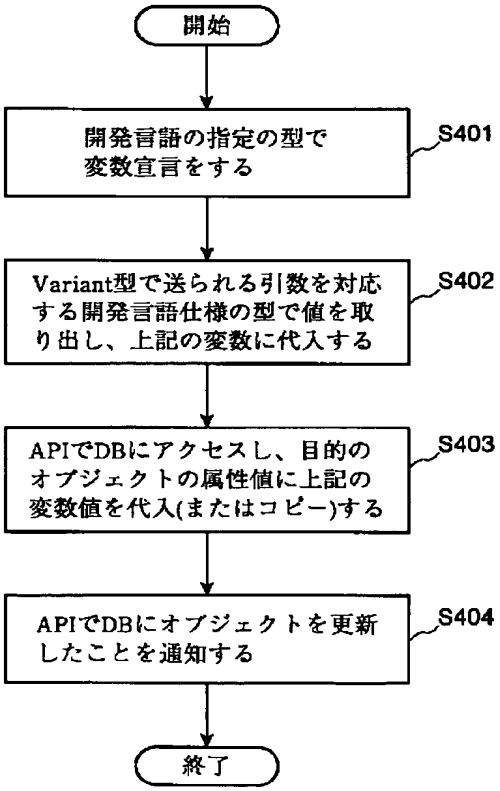
【図１】


クライアント
2
クライアント
プログラム
21
リモート
オートメーション
プロキシ
22
サーバ
1
サーバプログラム
13
オートメーションサーバ
14
リモート
オートメーション
マネージャ
15
データベース
11
API
12
自動生成
情報の取得
オートメーション通信
インターネット
5
プログラム自動生成装置
3
4
プログラム
テンプレート
ファイル

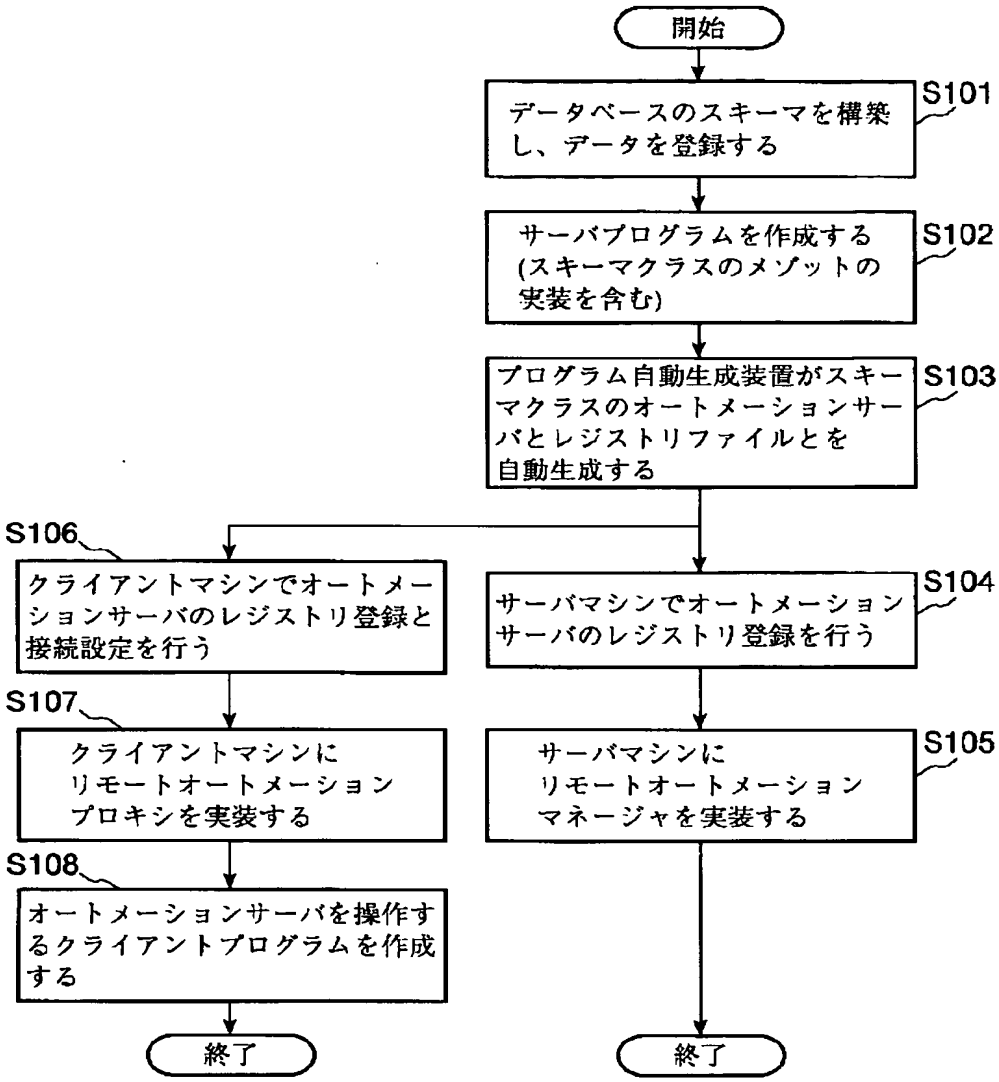
【図４】


開始
戻り値をVariant型で変数宣言し、指定の型を設定する
S301
APIでDBにアクセスし、目的のオブジェクトの属性値を取得する
S302
取得した属性値を最初に準備した変数に指定の型で代入し、関数の戻り値とする
S303
終了

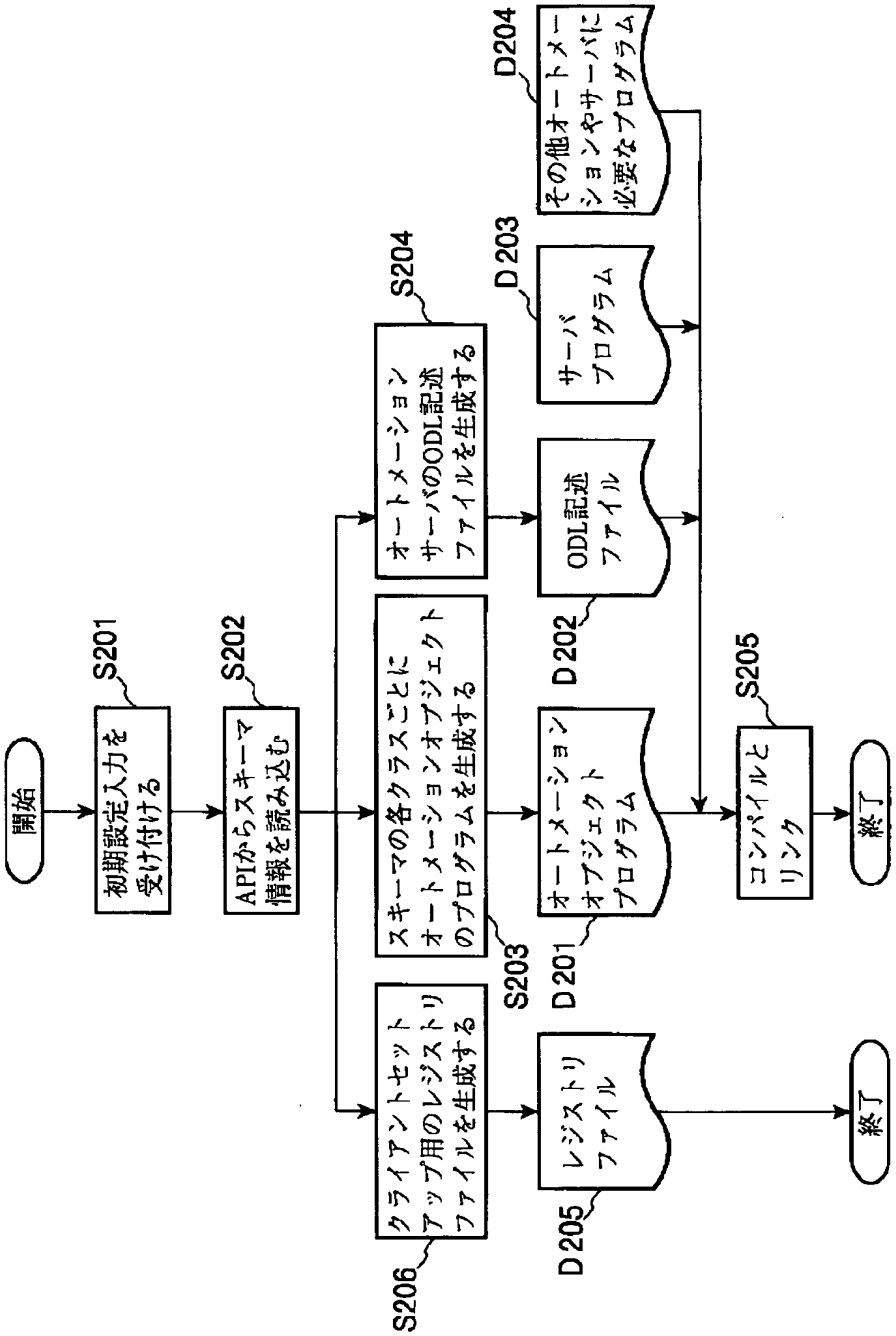
【図５】


開始
開発言語の指定の型で変数宣言をする
S401
Variant型で送られる引数を対応する開発言語仕様の型で値を取り出し、上記の変数に代入する
S402
APIでDBにアクセスし、目的のオブジェクトの属性値に上記の変数値を代入(またはコピー)する
S403
APIでDBにオブジェクトを更新したことを通知する
S404
終了

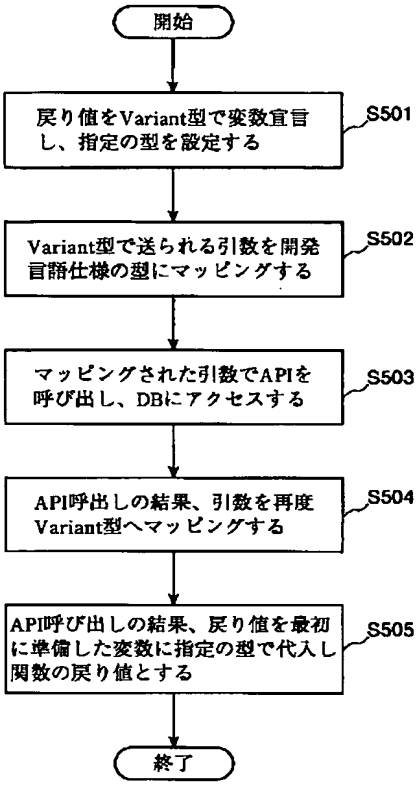
【図２】


開始
データベースのスキーマを構築し、データを登録する
S101
サーバプログラムを作成する(スキーマクラスのメソットの実装を含む)
S102
プログラム自動生成装置がスキーマクラスのオートメーションサーバとレジストリファイルとを自動生成する
S103
S106
クライアントマシンでオートメーションサーバのレジストリ登録と接続設定を行う
S104
サーバマシンでオートメーションサーバのレジストリ登録を行う
S107
クライアントマシンにリモートオートメーションプロキシを実装する
S105
サーバマシンにリモートオートメーションマネージャを実装する
S108
オートメーションサーバを操作するクライアントプログラムを作成する
終了
終了

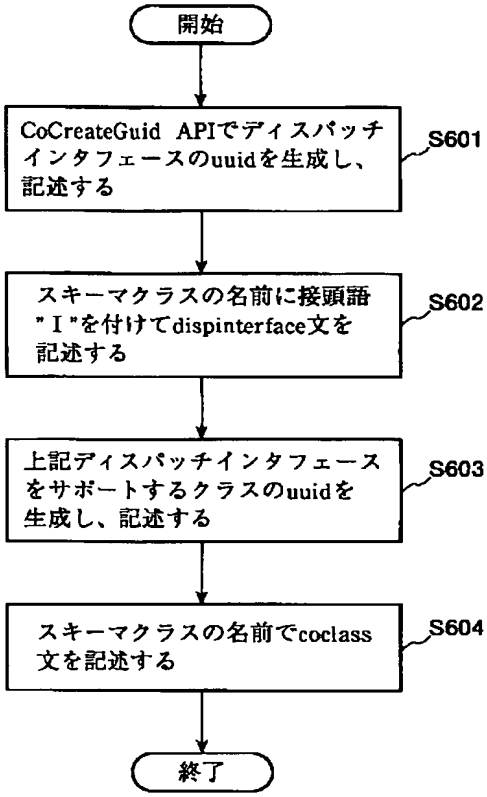
【図３】


開始
S201
初期設定入力を受け付ける
S202
APIからスキーマ情報を読み込む
スキーマの各クラスごとにオートメーションオブジェクトのプログラムを生成する
S203
S204
オートメーションサーバのODL記述ファイルを生成する
クライアントセットアップ用のレジストリファイルを生成する
S206
D201
オートメーションオブジェクトプログラム
D202
ODL記述ファイル
D203
サーバプログラム
D204
その他オートメーションやサーバに必要なプログラム
D205
レジストリファイル
S205
コンパイルとリンク
終了
終了

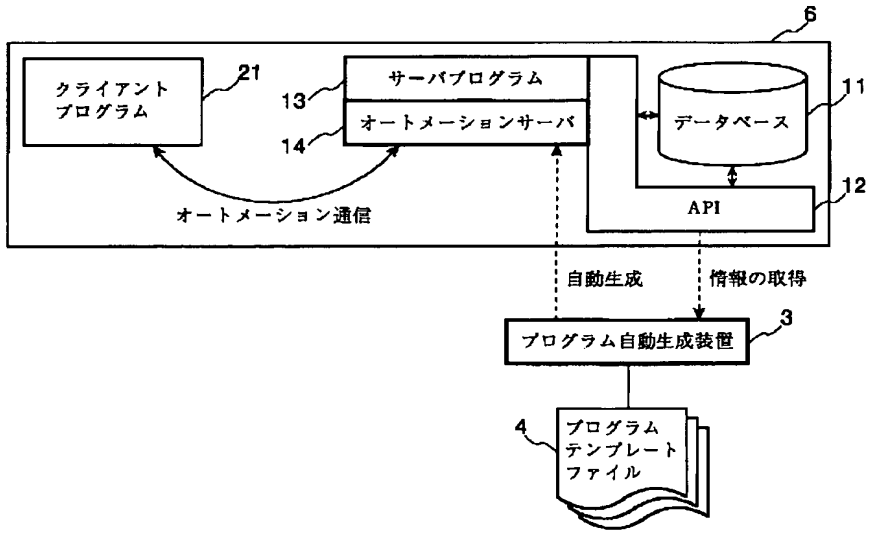
【図６】


開始
戻り値をVariant型で変数宣言し、指定の型を設定する
S501
Variant型で送られる引数を開発言語仕様の型にマッピングする
S502
マッピングされた引数でAPIを呼び出し、DBにアクセスする
S503
API呼出しの結果、引数を再度Variant型へマッピングする
S504
API呼び出しの結果、戻り値を最初に準備した変数に指定の型で代入し関数の戻り値とする
S505
終了

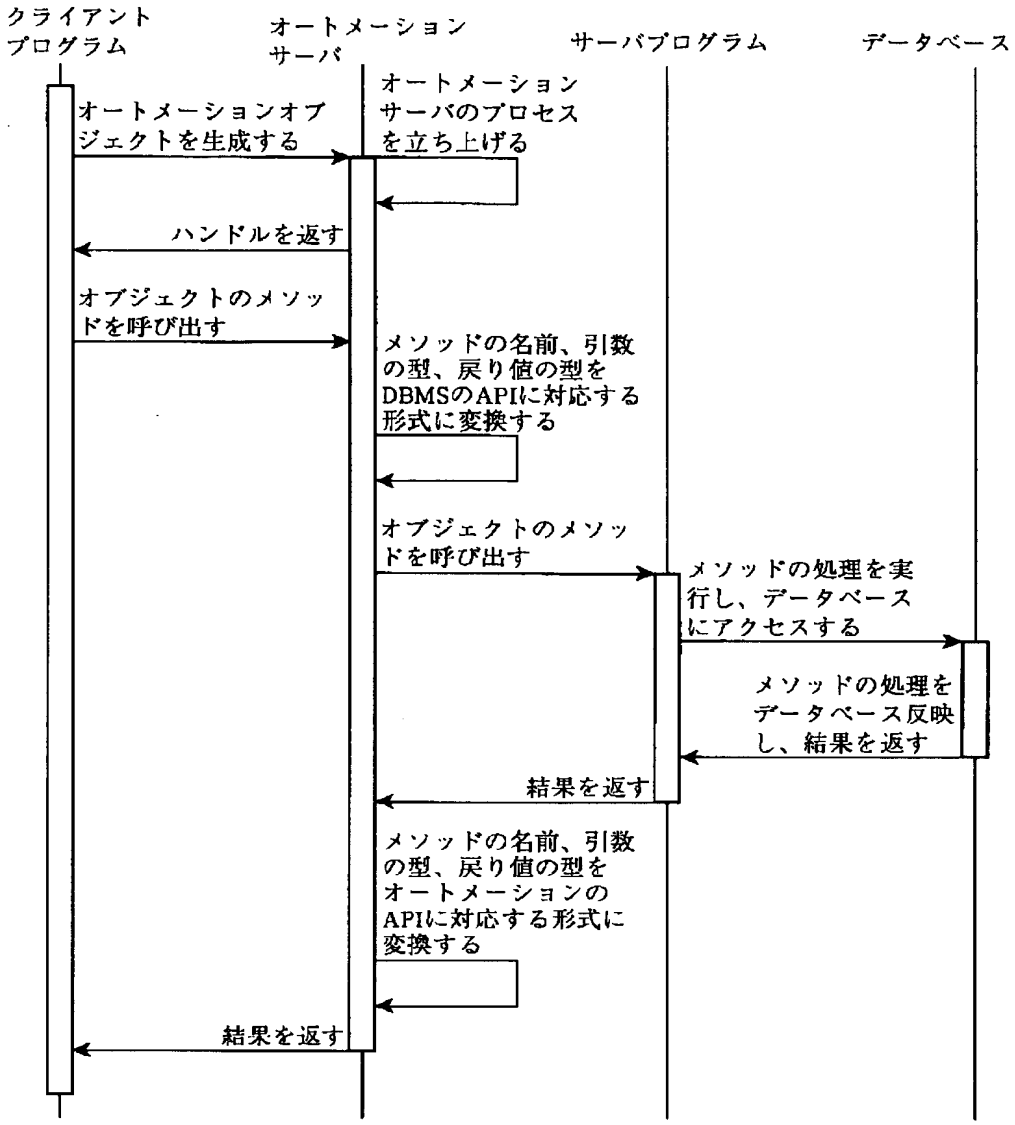
【図７】


開始
CoCreateGuid APIでディスパッチインタフェースのuuidを生成し、記述する
S601
スキーマクラスの名前に接頭語"Ｉ"を付けてdispinterface文を記述する
S602
上記ディスパッチインタフェースをサポートするクラスのuuidを生成し、記述する
S603
スキーマクラスの名前でcoclass文を記述する
S604
終了

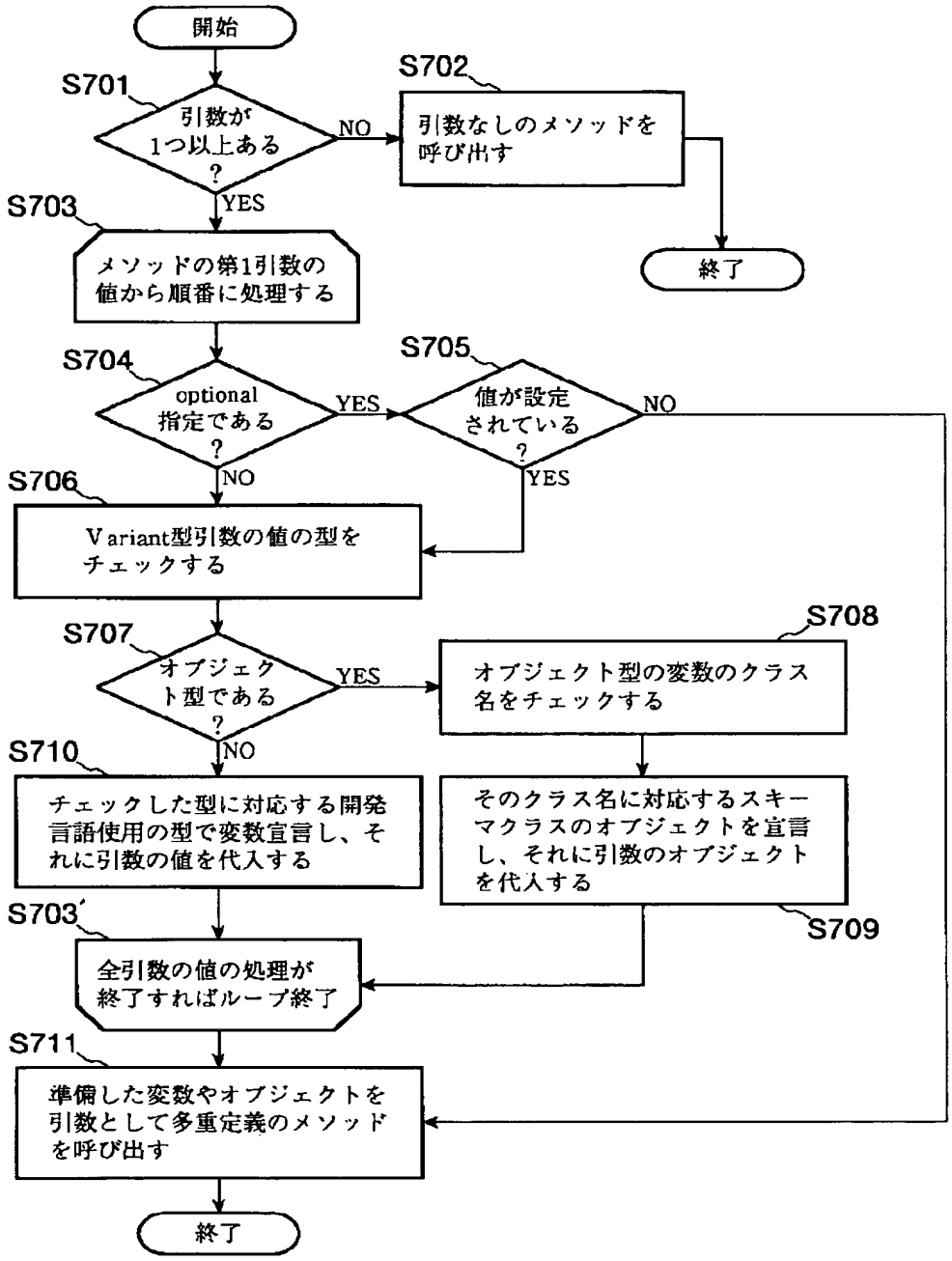
【図１１】


6
クライアントプログラム
21
13
サーバプログラム
14
オートメーションサーバ
データベース
11
API
12
オートメーション通信
自動生成
情報の取得
プログラム自動生成装置
3
4
プログラムテンプレートファイル

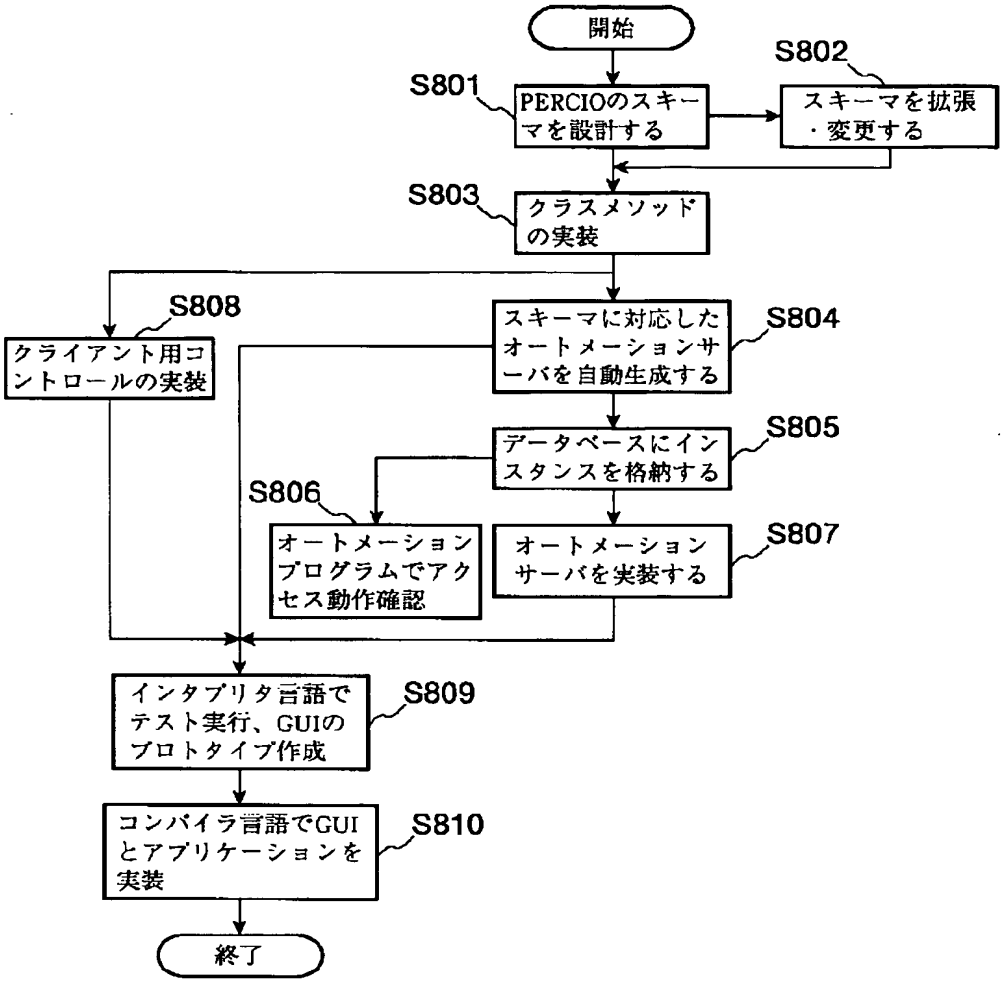
【図８】


クライアント
プログラム
オートメーション
サーバ
サーバプログラム
データベース
オートメーションオブ
ジェクトを生成する
オートメーション
サーバのプロセス
を立ち上げる
ハンドルを返す
オブジェクトのメソッ
ドを呼び出す
メソッドの名前、引数
の型、戻り値の型を
DBMSのAPIに対応する
形式に変換する
オブジェクトのメソッ
ドを呼び出す
メソッドの処理を実
行し、データベース
にアクセスする
メソッドの処理を
データベース反映
し、結果を返す
結果を返す
メソッドの名前、引数
の型、戻り値の型を
オートメーションの
APIに対応する形式に
変換する
結果を返す

【図９】


開始
S701
引数が1つ以上ある？
NO
S702
引数なしのメソッドを呼び出す
終了
YES
S703
メソッドの第1引数の値から順番に処理する
S704
optional指定である？
YES
S705
値が設定されている？
NO
YES
NO
S706
Variant型引数の値の型をチェックする
S707
オブジェクト型である？
YES
S708
オブジェクト型の変数のクラス名をチェックする
NO
S710
チェックした型に対応する開発言語使用の型で変数宣言し、それに引数の値を代入する
そのクラス名に対応するスキーマクラスのオブジェクトを宣言し、それに引数のオブジェクトを代入する
S709
S703'
全引数の値の処理が終了すればループ終了
S711
準備した変数やオブジェクトを引数として多重定義のメソッドを呼び出す
終了

【図１０】


開始
S801
PERCIOのスキーマを設計する
S802
スキーマを拡張・変更する
S803
クラスメソッドの実装
S808
クライアント用コントロールの実装
スキーマに対応したオートメーションサーバを自動生成する
S804
データベースにインスタンスを格納する
S805
S806
オートメーションプログラムでアクセス動作確認
オートメーションサーバを実装する
S807
インタプリタ言語でテスト実行、GUIのプロトタイプ作成
S809
コンパイラ言語でGUIとアプリケーションを実装
S810
終了